Guiding the Service Engineering Process: the Importance of Service Aspects

Gu, Q.; Lago, P.; Di Nitto, E.

published in
IFIP Workshop on Enterprise Interoperability
2009

document version
Publisher's PDF, also known as Version of record

Guiding the Service Engineering Process: The Importance of Service Aspects

Qing Gu[1], Patricia Lago[1], and Elisabetta Di Nitto[2]

[1] Dept. of Computer Science, VU University Amsterdam, The Netherlands
{qinggu,patricia}@cs.vu.nl
[2] Dip. di Elettronica e Informazione, Politecnico di Milano, Italy
dinitto@elet.polimi.it

Abstract. Service-oriented System engineering (SOSE) and traditional software engineering mainly differ for their focus and aims. These differences are reflected by a number of aspects peculiar to SOSE (*service aspects*). In this paper we specifically discuss three service aspects: *the relevance of cross-organizational collaboration, increased importance of the identification of stakeholders*, and *the need for increased effort at run-/change time*. We argue that SOSE methodologies provide better guidance on their application when service aspects are emphasized in associated process models. By highlighting the three service aspects in a process model of the methodology defined in a large European project, we show specifically how each aspect provides guidance for engineering service-oriented systems in practice.

Keywords: Service-oriented system engineering, SOSE methodology, Process model, Service aspects.

1 Introduction

Service-oriented systems are constructed by integrating heterogeneous services that are developed using various programming languages and running on heterogeneous operating systems from a range of service providers [1]. The engineering of such systems is different from Traditional Software Engineering (TSE) in that: the *focus* is shifted from engineering applications to developing compositions of services; the *control of services* is passed from their users to other owners (i.e. users of services do not have the control of them), and the *aims* are not only to satisfy required functionality and quality (e.g. performance, security, maintainability) but also to have the ability of adapting to ever-changing requirements (e.g. flexibility, dynamicity).

Many SOSE methodologies have been proposed in both academia and industry aiming at providing approaches, methods and (sometimes) tools for researchers and practitioners to engineer service-oriented systems (see for instance [2]). However, without being fully understood, a methodology is less valuable no matter how perfect it is. This is particularly relevant to SOSE methodologies as they are more complex than TSE ones, having to deal with new challenges while keeping the principles of TSE. The additional complexity results mainly from open

R. Poler, M. van Sinderen, and R. Sanchis (Eds.): IWEI 2009, LNBIP 38, pp. 80–93, 2009.

world assumptions, co-existence of many stakeholders with conflicting requirements and the demand of adaptable systems [3].

To improve the understandability of a methodology and its guidance, software development process models (describing what activities a development process consists of and how they should be performed) have been often used since they visualize the development process proposed by the methodology. The role of process models is clearly identified in a survey [4] of leading model-based system engineering methodologies, conducted by the INCOSE[1] community.

The service engineering community has realized that traditional software process modeling techniques are no longer directly applicable or adaptable in SOSE [5]. To overcome the mismatch between traditional software process models and SOSE, a number of service life cycle models have been proposed by both industry and academia (e.g. [6,5,2,7]). However, none of the proposed models has either reached a sufficient level of maturity or been able to fully express the aspects that are peculiar to SOSE. Besides different names on the phases and on the stakeholders, one might wonder what the real difference between these models and many well defined and experimented TSE approaches is.

Service aspects are issues that are specifically relevant to SOSE. These aspects reveal the core distinctions between the service-oriented paradigm and the traditional ones (e.g., component-based paradigm). Accordingly, the implication of service aspects should be explicitly expressed in SOSE process models. For instance, due to the dynamic nature of service-oriented systems, service artifacts (e.g. service specifications, service level agreements) are often generated on the fly and used dynamically, whereas artifacts in TSE (e.g. requirements specifications) are produced in a more static way, often within one single organization. Furthermore, service artifacts like service specifications are no longer limited to local use; rather, they can be published, discovered and reused across various SOSE projects [8] and activities scattered across multiple enterprises. As a result, SOSE pays particular attention to the way loosely related activities contribute to cross-organizational collaboration. Therefore, we argue that cross-organizational collaboration should be specifically expressed in SOSE process models to improve the guidance of applying SOSE methodologies.

In our previous work [7] we identified three service aspects that are crucial to the SOSE development process, namely a) the relevance of cross-organizational collaboration, b) the importance of the identification of stakeholders, and c) the need for more effort at run- and change time. We also defined a stakeholder-driven approach that illustrates such service aspects in a SOSE process model developed from the literature.

Here we further build upon the previous work. In particular, we refine and detail the service aspects and stress the necessity of expressing them in a SOSE process model. With the aim of validating the service aspects and their relevance to SOSE, we modeled the methodology developed and used in the SeCSE[2] European project. This has allowed us to: 1) build on concrete examples that

[1] www.incose.org/

[2] www.secse-project.eu

emphasize the relevance of the service aspects, and 2) show the applicability of the approach in practice. The results of this case study show that, by emphasizing service aspects in a SOSE process model, attention is naturally brought to those parts of the process model that are different as compared to TSE. The benefit is that guidance for applying a certain SOSE methodology is improved, and better service engineering management strategies can be put in place.

This work does not intend to propose a set of particular graphical notations for the purpose of modeling the SOSE development process. Instead, we intend to highlight *what* should be expressed in a SOSE process model. The graphical patterns (and associated notations) used in this paper illustrate one possible way of describing the service aspects expressively in a concrete SOSE process model.

The reminder of the paper is organized as follows. In Section 2, we explain the relevance of the service aspects to SOSE. In Section 3, we present the case study that we carried out and highlight the relevance of the three service aspects to the SeCSE methodology. Related work on the topic of SOSE process models is discussed in Section 4. Finally, Section 5 concludes the paper.

2 The Service Aspects

The fundamental change in developing service-oriented systems as opposed to traditional software systems is that software is delivered as a service. As such, users pay for and use services instead of buying and owning software. Consequently, users do not have the control of services, which are owned and controlled by service providers instead. These changes are reflected by three service aspects identified in [7]. In this section, we refine and detail these service aspects and stress the necessity of expressing them in a SOSE process model.

2.1 The Relevance of Cross-Organizational Collaboration

The focus of SOSE is shifted from applications to services that are collaboratively developed by multiple SOA roles [9,10], such as service consumer, service provider, service broker. During the development process, activities like specification&modeling, design, implementation, testing, operation and maintenance are all required to be performed in a collaborative manner [11].

For instance, service-oriented systems are built through discovering and composing existing services from multiple service providers rather than coding as in TSE. Consequently, the processes of discovering, selecting, composing services require continuous interaction (or *collaboration*) between the participating roles through the service development life cycle. Hence, collaboration between participating roles becomes *explicit* and *critical* in that it enters the details of a SOSE process that is now scattered across multiple roles. This makes their relationship tighter but also demanding clearer governance and agreements.

What makes it more critical is that these roles are often distributed in multiple departments or organizations. In this case, interactions between development

activities associated with multiple business roles demand for collaboration between multiple organizations. We call this type of collaboration, which crosses the boundaries of the domain of each role, *cross-organizational collaboration*.

In TSE, and especially in concurrent software development, component-based software development, or outsourcing, cross-organizational collaboration also occurs when one organization delegates a set of tasks to the other organization(s). The main difference is that in TSE the first organization only concerns how software is developed internally, but not how delegated tasks are carried out externally. Consequently, only the results of the delegated tasks are of importance to the development process of the first organization. As a result, the cross-organizational collaboration is a purely *buying (outsourcer) and selling (supplier)* relationship, and its details are hidden from the perspective of the development process of the first organization. In SOSE, instead, the collaborative roles coexist in a service-oriented system rather than having an active-passive relationship. Detailed examples of the way in which coexisting roles collaborate are further discussed in Section 3.2.

When collaboration crosses the boundaries of each organization, barriers (e.g., conceptual, technological barriers, and organizational barriers) to enterprise interoperability often obstruct the effectiveness of collaboration. Since collaboration between multiple roles becomes part of the SOSE process, it is of great importance to highlight this collaboration in a SOSE process model. When the collaboration becomes explicit and clear, the need for corresponding agreements or contracts becomes evident. Consequently, appropriate governance can be applied. As such, barriers to enterprise interoperability can be reduced.

2.2 Increased Importance of the Identification of Stakeholders

A *stakeholder* can be defined as a person, group or organization playing a well defined role (or roles) in a SOSE methodology. Since cross-organizational collaboration becomes more critical in SOSE, the importance of clearly identifying stakeholders increases accordingly. If stakeholders are identified at a too coarse granularity, the represented interaction remains not fully specified. This leads to unclear responsibilities among collaborating enterprises and thus decrease in trust and possibly in success. Because the level of details matters, the identification of stakeholders directly determines the level of detail expressed in a SOSE process model.

The decision on whether a role should be identified as a separate stakeholder in a SOSE process model depends on what type of interactions the model intends to represent. For instance, if a SOSE development approach intends to emphasize or elaborate on how service monitoring is provided, accordingly, service monitor could be selected as a stakeholder in a SOSE process model. As such, it offers the possibility to explicitly associate monitoring-related activities with the service monitor and to explicitly describe the interaction between the service monitor and other stakeholders. Of course, if service monitoring is not the main focus, then it is not necessary to select it as a separate stakeholder since

the detailed interaction between a service monitor and the other stakeholders is not of interest.

As a general rule, if service monitoring (or any other activity) is performed by an independent third party, the corresponding role should better be identified as a separate stakeholder because it stands for an external organization. As such, it offers the possibility to explicitly express the responsibilities across different business domains.

The importance of the identification of stakeholders in SOSE process models also lies in the fact that stakeholders in the SOSE development process do not always assume the same roles as in TSE. For instance, in general a software developer is responsible for coding or implementing software applications, while in SOSE a service developer could be responsible also for composing existing services, depending on specific methodologies. Without specifically associating SOSE activities to stakeholders, one is not able to visualize the corresponding responsibilities as one would do in TSE.

In summary, identifying stakeholders with the appropriate level of detail in a SOSE development approach facilitates the establishment of a corresponding SOSE process model describing associated activities and their interactions at an appropriate level of abstraction.

2.3 The Need for Increased Effort at Run-/Change Time

In TSE, the main goal is to develop high quality applications that meet the requirements of the end users. Consequently, most of the effort is dedicated to design (collecting requirements, design, and implementation) and change time (maintenance). Runtime activities are hardly addressed if not in specific domains. Furthermore, change time activities are often performed off line (either with or without execution interruption).

Different than TSE, the main goal of SOSE is not only to deliver high quality but also agile and robust services which are able to meet the *ever-changing* business requirements. Consequently, much more development effort is shifting from design time to run-/change time. For instance, components identification is often performed at design time in TSE; the SOSE equivalent activity is service discovery, which is encouraged to be performed at runtime and it is regarded as one of the major challenges in the SOSE field.

As discussed in Section 4, most existing SOSE process models do fail in emphasizing this shift. By explicitly modeling the two stages, a process model can visualize the amount of activities shifted to run-/change time, hence providing useful inputs to resource allocation.

3 Applying Service Aspects to a Concrete Methodology

With the aim of gaining insight in the extent to which the modeling of the three service aspects improve the guidance of a SOSE development approach, we modeled them in a concrete and practical context, i.e. the SeCSE methodology [12].

In this section, we first introduce the SeCSE methodology and present its process model (with the three service aspects being highlighted), followed by a discussion of how each service aspect both emphasizes the characteristics of the SeCSE methodology itself and facilitates better SOSE guidance.

3.1 The SeCSE Methodology

The SeCSE project is a EU-funded project. Its goal was to investigate methods, techniques and tools to develop and manage service-oriented systems in an effective way. A large number of academic and industrial partners have been collaborating in this project. As a consequence, the resulting SeCSE methodology has both theoretical and practical value.

The SeCSE methodology describes the main development activities and tools that have been adopted by the SeCSE project. It provides ways to create service compositions where component services are discovered at runtime either on the basis of the context of usage or when a certain service fails. This focus on runtime is one step forward towards the third generation service-oriented systems [13].

Although service discovery is regarded as one of the major activities in developing service-oriented systems, and even though techniques already exist to support service discovery, in practice service discovery is hardly adopted. Nowadays, most enterprises focus on migrating legacy systems to service-oriented systems and implementing new services rather than discovering services from a registry (as service-oriented systems are supposed to do). In the SeCSE project, service discovery is not only addressed by the SeCSE methodology, but also experimented in the consortium. As an advanced and relatively mature approach, the SeCSE methodology is a good candidate to be selected as the case study in this work to analyze the service aspects addressed by it.

Moreover, the design of the SeCSE methodology does not specifically or consciously take our three service aspects into consideration. This provides us the possibility to take the SeCSE methodology as such and model the service aspects addressed by the methodology. Comparing to the original process model illustrated in the documentation of the SeCSE methodology, the SeCSE process model proposed in this work provides better guidance for its users.

3.2 The SOSE Process Model for the SeCSE Methodology

By focusing on service aspects, our main objective is to discuss *what* has to be modeled rather than *how* to model. For illustration purposes, we use BPMN[3] as process modeling notation to be used to communicate a methodology to its users. This is expressive enough to represent the various inter-dependencies and multiple stakeholders involved in the SeCSE development process.

For the purpose of modeling the service aspects of the SeCSE methodology, we illustrated the SeCSE development process by means of a process model. The decisions and assumptions that we have made to construct the SeCSE process

[3] www.omg.org/spec/BPMN/

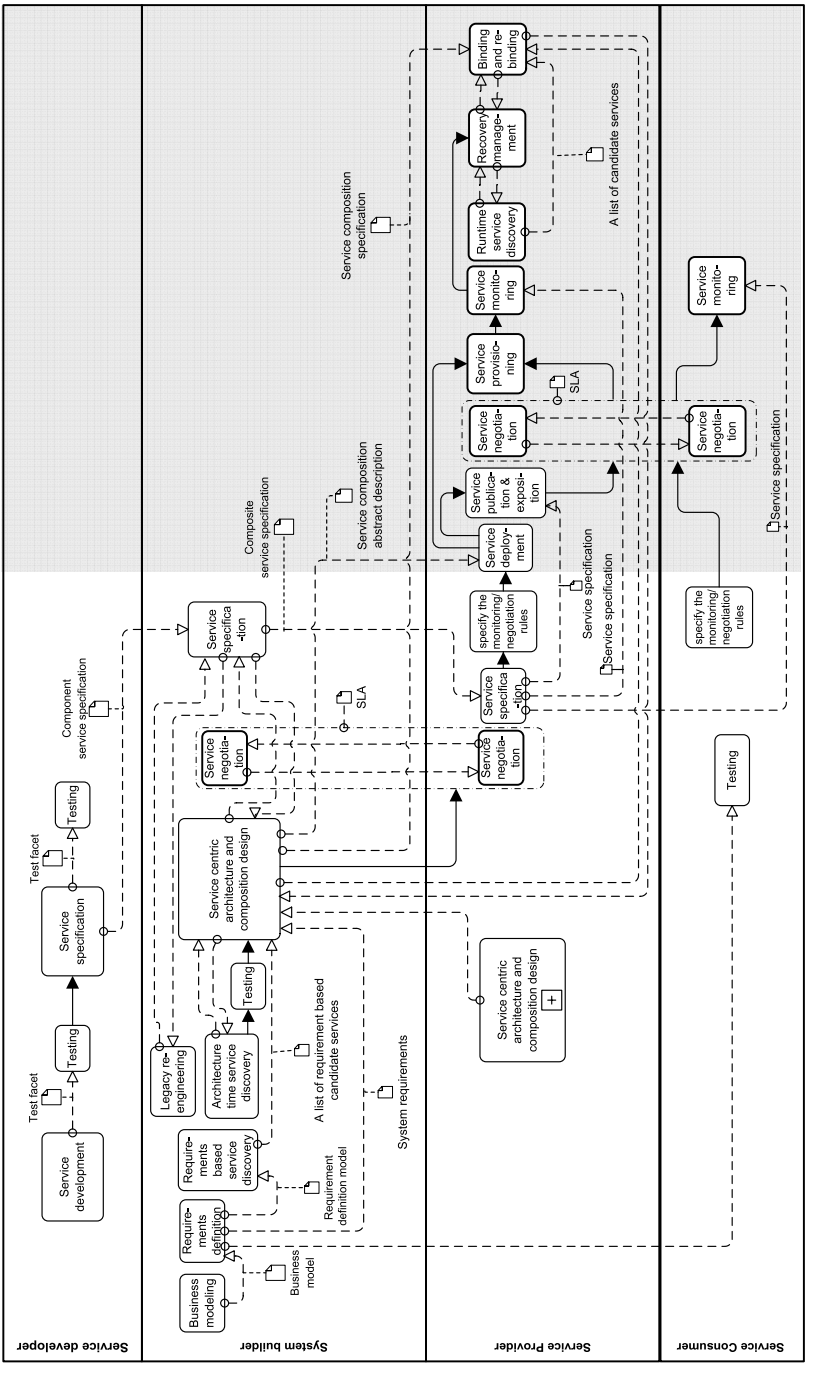

Fig. 1. The SOSE process model for the SeCSE methodology

model were verified with the SeCSE experts to check for correspondence of the model to the methodology. This analysis has also helped to elicit information that were missing or left implicit in the methodology document. After the verification from the SeCSE experts, we were able to refine and finalize the model[4].

The resulting model is given in Fig. 1. In this model, we specifically modeled the stakeholders identified in the SeCSE methodology to highlight the *increased importance of the identification of stakeholders*; we specifically modeled their associated activities and inter-dependencies to highlight the *cross-organizational collaboration*. The *increased effort at run-/change time* becomes obvious in the model since we separated them from the design time effort.

For each service aspect we first explain its associated graphical pattern in a SOSE process model in general; and then we discuss how the SeCSE methodology addresses the service aspect by observing the SeCSE process model against the graphical pattern.

The Relevance of Cross-Organizational Collaboration. Fig. 2 graphically illustrates the cross-organizational collaboration (COC) service aspect. The left-hand side of the figure shows three collaboration types (COC patterns): *peer activities group*, *main-sub activities* and *distributed activities*. These patterns are exemplified in the right-hand side of Fig. 2.

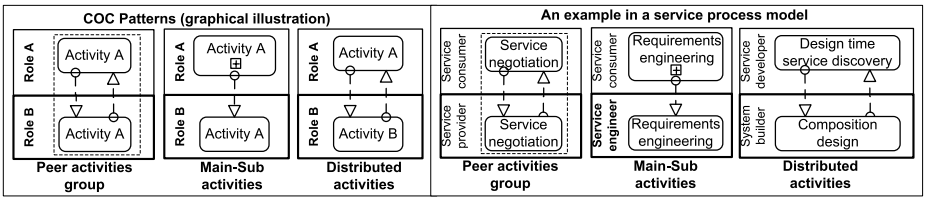

Fig. 2. Service aspect: cross-organizational collaboration (COC)

- The **peer activities group** models same activities carried out *in parallel* across multiple partner enterprises. For instance, service negotiation can be carried out by a service provider and a service consumer with the common objective of reaching the agreement of service provision and consumption.
- The **main-sub activities** model the same activities carried out *partially by one partner enterprise and completed by another*. For instance, a service consumer may perform the user-centric part of a requirement engineering activity that the service provider is mainly responsible for.
- The **distributed activities** model *inter-dependent activities carried out across multiple partner enterprises*. For instance, design time service discovery can be carried out by a service developer and composition design can be carried out by a system builder. The former provides input (such as discovered candidate services) to the latter; the latter might also provide feedback

[4] Due to the limited space, the design of the case study itself is not described in detail.

to the former when different design decisions are taken, possibly requiring different service candidates.

By observing the SeCSE process model (shown in Fig. 1) against the three COC patterns defined in Fig. 2, the attention is brought to specific types of cross-organizational collaboration.

- **Peer activities group:** Service negotiation occurs twice in the SeCSE development process. One is carried out by a service provider and a service consumer; another is carried out by a system builder and a service provider. By nature, each service negotiation must be performed in parallel by both its stakeholders as peers. The results of the collaborations (indicated by the data objects attached to the peers) are SLAs. Different from TSE where contracts are often established after software is built, SLAs in SOSE often precede final service products (service composition in the case of the SeCSE methodology). These SLAs are also potentially useful to other activities such as service monitoring.
- **Main-sub activities:** Service centric architecture and composition design are carried out by a system builder and service provider in a cooperative manner. A system builder has the main responsibility for this activity, whereas a service provider focuses only on a subset of its tasks. For instance, the service provider might work on the definition of the list of possible candidate services to be used at runtime; while the system builder is responsible for the overall service composition design. In this way, the subtasks that the service provider takes are of competence of the system builder.

  Service specifications are modeled as three activities with related service specification as data objects. They are carried out by a service developer, system builder and service provider independently but on related artifacts. In general, a service developer creates service specifications for a component service, which influences a composite service carried out by a system builder. The system builder has to make sure that the QoS characteristics defined in the specification of the component services are compatible with those of the composite service. When a service composition or a single service is deployed, the service provider may add information to the corresponding specification known at deployment time.
- **Distributed activities:** Service centric architecture and composition design is carried out by a system builder at design time and binding and re-binding is carried out by a service provider at run-/change time. Cross-organizational collaboration occurs when new substituting services are discovered at run time (e.g., due to a new requirement) and service composition needs to update its bindings to accommodate the change.

Only when the collaboration is explicitly captured, the stakeholders of service-oriented systems can gain insight on the impact between their own responsibilities and the others'. Each stakeholder has a clearer view on at what time (*"when"*) which activity (*"what"*) has to be carried out in cooperation with which stakeholder (*"who"*) and in which manner (*"how"*). In the SOSE development process,

external enterprises often continuously play important roles throughout the service life cycle. By looking for the cross-organizational collaboration patterns in a service process model, enterprises are brought to focus on the points needing strategic business agreements that should regulate such tight collaboration.

Increased Importance of the Identification of Stakeholders. Fig. 3 graphically represents the stakeholders pattern (and an example), which makes the stakeholders in a SOSE process model explicit. By observing the SeCSE process model (shown in Fig. 1) against this stakeholder pattern, we can see that the SeCSE methodology involves mainly four stakeholders, namely: service developer, system builder, service provider and service consumer). These stakeholders, potentially representing partner enterprises, play common SOA roles from the perspective of service implementation, integration provision, and consumption.

Explicitly modeling the identified stakeholders improves the guidance of the SeCSE methodology as follows. Firstly, by placing the SOSE activities in the corresponding swimlanes, the SeCSE process model naturally shows the responsibilities and collaborations of and among stakeholders. This is especially crucial in SOSE where cross-organizational collaboration occurs in almost all activities. In this way, the business dependencies requiring contractual/SLA agreements are made explicit, and project managers can better plan the allocation of development activities based on the skills and responsibilities of the internal and external stakeholders.

Secondly, service composition centered characteristic of the SeCSE methodology is well captured by the SeCSE process model when identified stakeholders are explicitly modeled. Fig. 1 shows that most of the development activities are associated to the system builder and the service provider (stakeholders that carry out service composition activities). Furthermore, the model shows that the system builder and the service provider are tightly linked; the service developer and the service consumer are instead loosely linked. Due to focus of service composition, the service consumer in the SeCSE process model is considered as the consumer of composite services, rather than the consumer of component services. Therefore, the service consumer does not have direct interaction with the service developer, and the system builder must cooperate with the service provider in multiple activities. In this way, the SeCSE process model very well captures the fact that service composition is the main focus of the SeCSE methodology and consequently provides better guidance in that the stakeholders are able to gain better understanding of the focus of the (SeCSE) methodology.

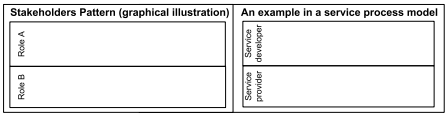

Fig. 3. Service aspect: increased importance of the identification of stakeholders

The Need for Increased Effort at Run-/Change Time. Our approach of separating the design and run-/change time activities in a SOSE process model is presented in Fig 4, where the left-hand side of the figure shows the 2-stages pattern, exemplified in the right-hand side of the figure. In this example, it is visually evident that service design is carried out at design time; while service discovery is performed at run-/change time.

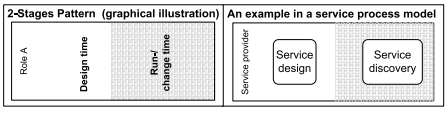

Fig. 4. Service aspect: increased effort at run-/change time (2-Stages)

By observing the SeCSE process model (shown in Fig. 1) against the 2-stages pattern defined in Fig. 4, we are now able to easily distinguish the design time activities (falling in the left-hand side of the figure) from the run-/change time activities (in the shadowed area at the right-hand side of the figure). Consequently, the guidance for applying the SeCSE methodology is improved in that the process model shows its support for adaptation, service composition and facilitates critical project plan decisions.

Firstly, we notice that about one third of the development effort is dedicated to run-/change time activities. In particular, runtime service discovery and service negotiation are supported by the SeCSE methodology with the objective of increasing the adaptability and agility of resulting systems to meet on-the-fly requirements. Thereby, related activities such as runtime service monitoring, recovery management, and binding and re-binding are also in place.

Secondly, we notice that the development effort dedicated to run-/change time activities is not evenly distributed among the stakeholders in the SeCSE methodology. Instead, the service provider carries out most of the run-/change time activities; while the system builder and service developer do not perform run-/change time activities at all. We have discussed in Section 3.2 that the roles of system builder and service provider are extensively developed due to the service composition centered approach. The process model illustrates and emphasizes further the separation of design and run-/change time activities: the system builder focuses on the design of service compositions; while the service provider focuses on the provision of service compositions.

Thirdly, knowing which activities are executed at which stage is also crucial in SOSE. Project managers should be able to adjust project plans based on the criticality of the activities since runtime activities are directly related to executing services real-time and therefore more critical than design time activities. For instance, as shown in Fig. 1 service negotiation is supported by the SeCSE methodology at both design time and runtime. The difference is that at design time, service negotiation occurs between a system builder and a service provider for component services that are selected for service composition;

at runtime it occurs between a service provider and a service consumer for a composite service that fulfills business requirements. This difference results in different levels of business commitment. At design time, the failure of reaching service level agreements or the failure of collaborating with a service provider does not have huge business impact on the system builder; the system builder can always decide to look for an alternative service. However, at runtime if the composite service fails to execute or does not reach the quality it promises, the system builder faces risks to loose its customer and even its business market. Here, the business commitment is much higher than at design time. Being aware of this difference, a project manager is able to decide which actions to take for activities with various levels of criticality.

Summary. As auspicated, highlighting the service aspects in the SeCSE process model allows those who have to exploit the methodology to have more clear evidence of issues (cooperation between organizations, numerous stakeholders, activities to be executed during the operaton phase) that are critical from the managerial point of view. By observing the service aspects captured in the model, we may conclude (and becomes more evident) that some differences between SOSE and TSE become obvious in the applied SeCSE context. Firstly, the fact that each of the four identified stakeholders is responsible for a common SOA role (centered on services) reflects the shifted focus from applications to service pools. Secondly, due to the fact that consumers of services do not have the control of them, more interactions between stakeholders (cross-organizational collaboration) occur, as shown by the many arrows crossing the various swimlanes. Thirdly, around one third of activities are carried out at run-/change time, which shows that resulting systems are dynamic and therefore have the potential ability to adapt to ever-changing requirements.

4 Related Work

It has been gradually recognized that traditional software process models are no longer sufficient to model the SOSE development process. To overcome the mismatch between traditional process models and the SOSE development process, a number of SOSE-specific process models have been proposed by both industry and academia. However, as we already discussed in [7], these assume that the development of service-oriented systems is entirely internal to an organization. For instance, the model proposed by IBM in [6] describes four phases that are implicitly assumed to be executed by IBM itself or any organization adopting the IBM methodology. This results in the fact that the actual difference between SOSE and TSE methodologies remain unclear. As discussed in Section 2, we argue instead that interactions across the organizational boundaries require in SOSE particular attention, and should be made explicit.

While the approaches found in the literature are proposals of specific methodologies and lifecycles, our approach can be seen as a way to interpret and investigate different existing lifecycles. This has a value per se as it does not force

people to adopt a specific approach for developing and managing service-oriented systems, but it helps them to understand and make coherent all the methods they use in their common practice. In a similar line of research, Blake in [5] advocates the distinction between two key activities: service development and service-centric system management. While the first activity follows a quite traditional iterative process, service-centric system management is seen as much more dynamic of the traditional processes associated with the development and the operation of other kinds of software. In particular, as in SeCSE, runtime (sub)activities such as re-binding are identified. The importance of stakeholders is stressed, and a number of them is identified and assigned to the various activities of the lifecycle. We differentiate from this work as we highlight not only stakeholders and their activities, but also the interaction between these stakeholders and the artifacts they produce and exchange. As we have argued in the previous sections, in fact, clarifying these aspects help all roles involved in the lifecycle in better understanding the critical aspects of the lifecycle and in properly drive it toward the achievement of the project goal.

Bell [14] proposes the structure of a SOSE process model, which consists of timeline, events, seasons and disciplines. As we do, he uses timelines to indicate a sequence of development activities. Design and run-/change time activities in our approach correspond to seasons in Bells structure, while our development activities can be regarded as disciplines in Bells structure. The approach is also focusing on defining and classifying those events that have an impact on the lifecycle. While a differentiation between runtime and design time activities is presented, all runtime aspects are not described in detail. Also, the approach does not seem to stress the aspects related to the interaction between the stakeholders that we consider of paramount importance.

5 Conclusions

In this paper, we emphasize the importance of explicitly expressing in process models service aspects that are peculiar to SOSE. We argue that having these service aspects highlighted would provide better guidance on the SOSE development process. We have applied our service aspects on a concrete SOSE methodology. The results show that these service aspects help understanding the SOSE methodology when they are made explicit in an associated process model. Moreover, the use of the methodology and project management are also facilitated.

Further, these service aspects emphasize the SOSE support of a certain methodology. In this way, they help identifying if the methodology itself will deliver 'real' service-oriented systems. For instance, by analyzing these service aspects in the SeCSE methodology, we can see that: it involves the standard SOA roles; it covers the interaction among these roles; and it pays particular attention to run-/change time activities. We therefore argue that the service-oriented systems it delivers would potentially be dynamic, agile and have good alignment to business requirements.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE). IEEE Computer Society, Los Alamitos (2003)
2. Papazoglou, M.P., van den Heuvel, W.J.: Service-oriented design and development methodology. Int. J. Web Engineering and Technology (IJWET) 2(4), 412–442 (2006)
3. Baresi, L., Nitto, E.D., Ghezzi, C.: Toward open-world software: Issues and challenges. IEEE Computer Society 39(10), 36–43 (2006)
4. Estefan, J.A.: Survey of model-based systems engineering (MBSE) methodologies, Jet Propulsion Laboratory, California Institute of Technology, Pasedena, CA (2007)
5. Blake, M.B.: Decomposing composition: Service-Oriented software engineers. IEEE Software 24(6), 68–77 (2007)
6. McBride, G.: The role of SOA quality management in SOA service lifecycle management. developerWorks (2007)
7. Gu, Q., Lago, P.: A stakeholder-driven service life cycle model for SOA. In: IW-SOSWE 2007: 2nd international workshop on Service oriented software engineering, Dubrovnik, Croatia, pp. 1–7. ACM, New York (2007)
8. Tsai, W.T., Jin, Z., Wang, P., Wu, B.: Requirement engineering in service-oriented system engineering. In: ICEBE 2007. Proceedings of the IEEE International Conference on e-Business Engineering. IEEE Computer Society, Los Alamitos (2007)
9. Tsai, W.T., Wei, X., Paul, R., Chung, J.Y., Huang, Q., Chen, Y.: Service-oriented system engineering (SOSE) and its applications to embedded system development. Service Oriented Computing and Applications, 3–17 (2007)
10. Colombo, M.M., Nitto, E.D., Penta, M.D., Distante, D., Zuccala, M.: Speaking a common language: A conceptual model for describing service-oriented systems. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 48–60. Springer, Heidelberg (2005)
11. Tsai, W.T.: Service-oriented system engineering: A new paradigm. In: Service-Oriented System Engineering, Beijing, China, pp. 3–6 (2005)
12. Penta, M.D., Bastida, L., Sillitti, A., Baresi, L., Ripa, G., Melideo, M., Tilly, M., Spanoudakis, G., Maiden, N., Cruz, J.G., Hutchinson, J.: SeCSE - Service Centric System Engineering: an overview. At your service: Service Engineering in the Information Society Technologies Program (2009)
13. Fitzgerald, B., Olsson, C.M.: The software and services challenge. In: EY 7th Framework Programme, Contribution to the preparation of the Technology Pillar on "Software, Grids, Security and Dependability" (2006)
14. Bell, M.: Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. Wiley, Chichester (2008)